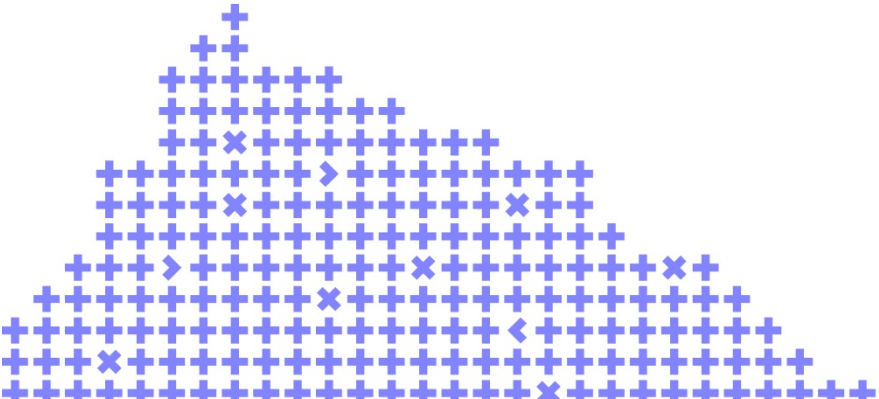
Developing a best-in-class deprecation strategy for your features or products

Addison Schultz



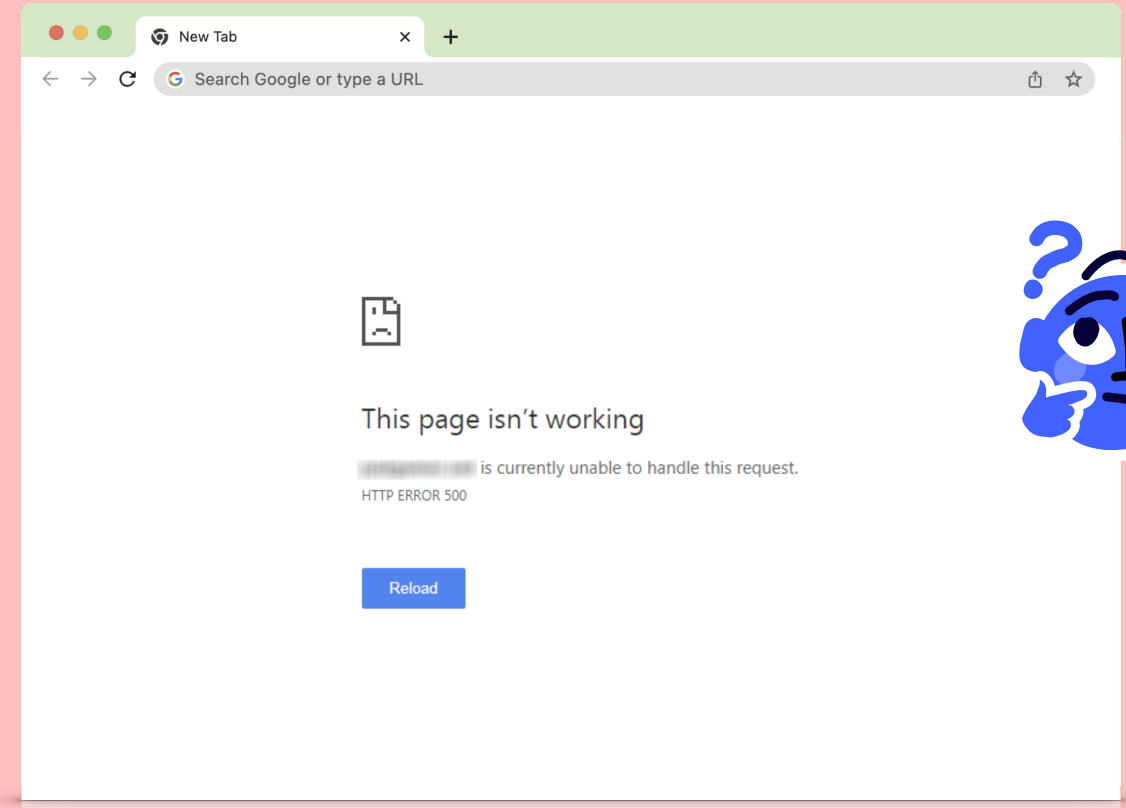
High Load ++ Armenia

Co-organizer

Yandex

Developing a best-in-class deprecation policy for your APIs

14 December 2022
HighLoad++ Armenia

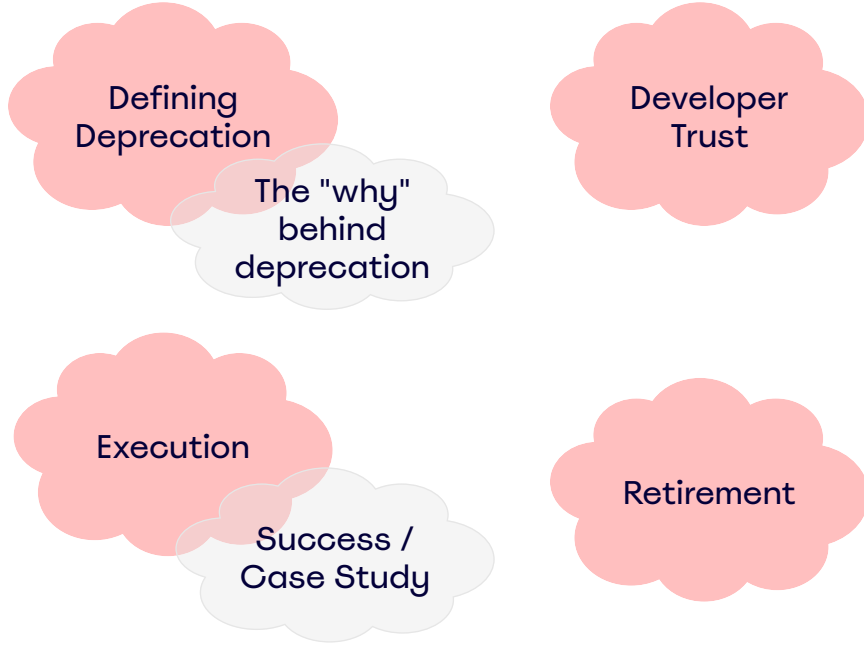

New Tab
Search Google or type a URL
This page isn't working
is currently unable to handle this request.
HTTP ERROR 500
Reload

miro

Addison Schultz

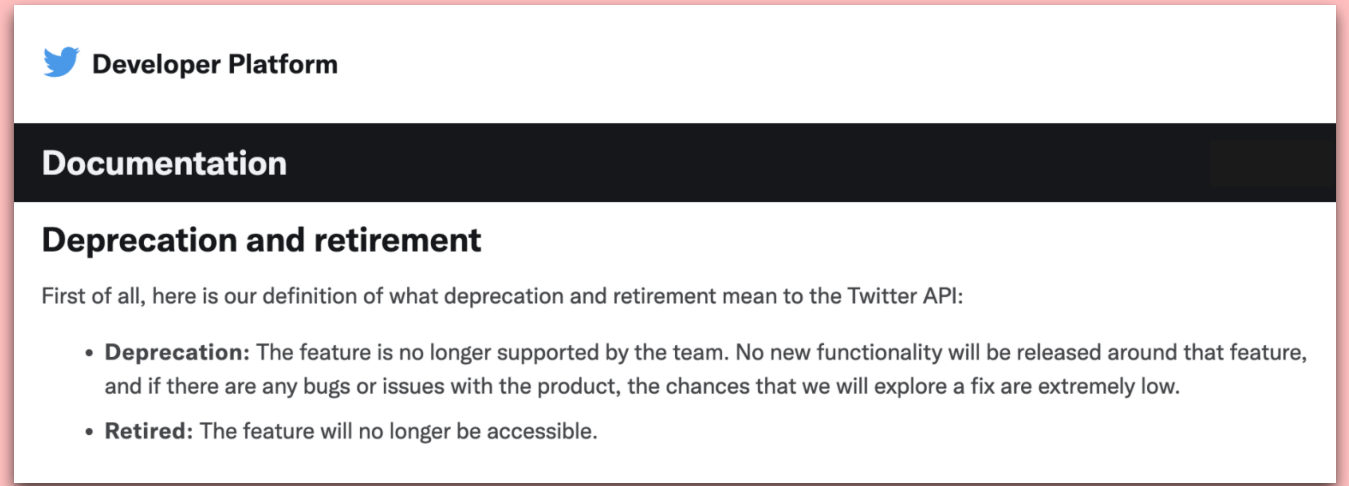
Developer Advocate @ Miro

miro


Defining Deprecation
The "why" behind deprecation
Developer Trust
Execution
Success / Case Study
Retirement

miro

What ~exactly~ is deprecation?



Developer Platform

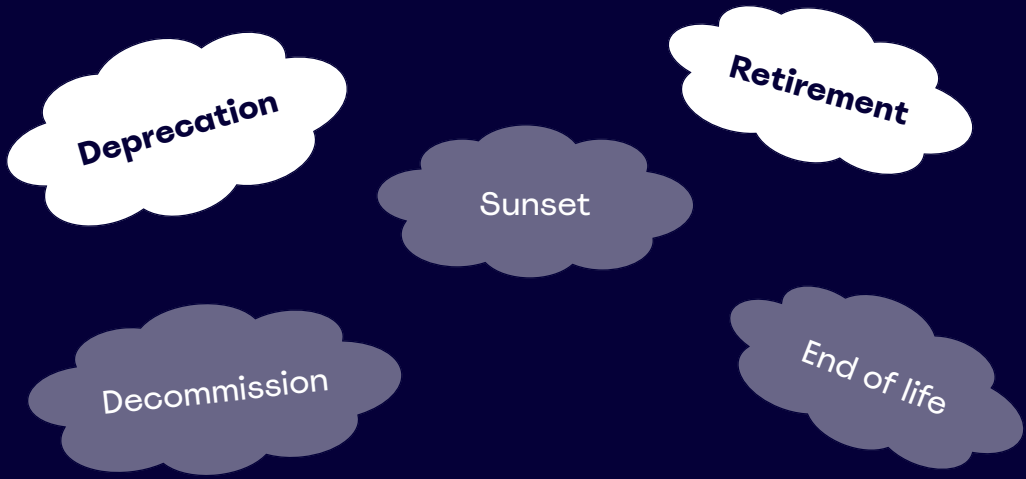
Documentation

Deprecation and retirement

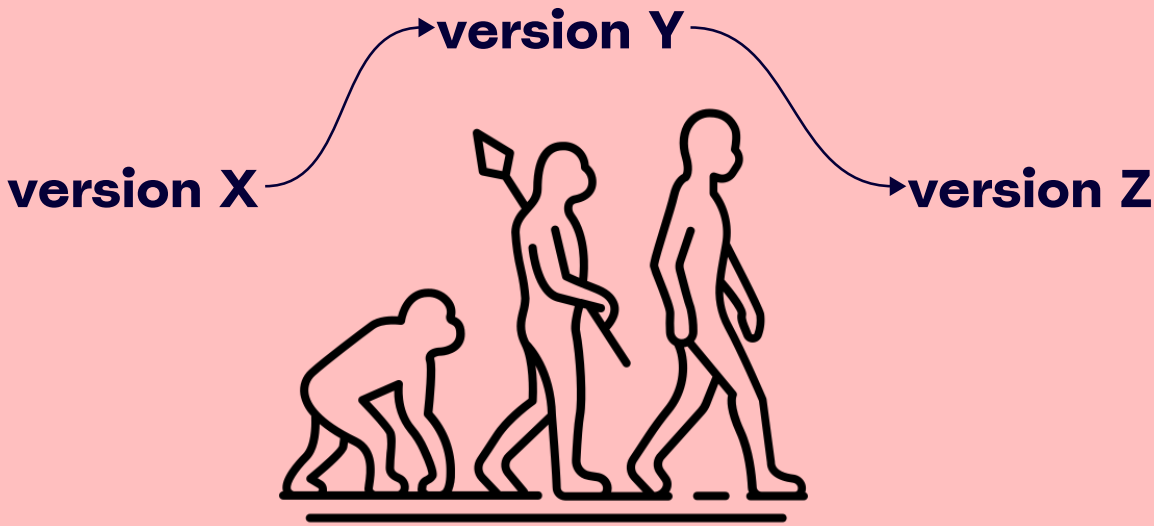
First of all, here is our definition of what deprecation and retirement mean to the Twitter API:

- **Deprecation:** The feature is no longer supported by the team. No new functionality will be released around that feature, and if there are any bugs or issues with the product, the chances that we will explore a fix are extremely low.
- **Retired:** The feature will no longer be accessible.

Terms vary, but the principles remain



The evolution of API capabilities



Trust is at the heart of deprecation

Loss of trust with
developers or end users
is your single biggest risk

Avoiding ambiguity

Unclear timelines

No internal consensus

Undocumented behavior

Your policy should build developer trust

Your APIs — 🤝 — Developers

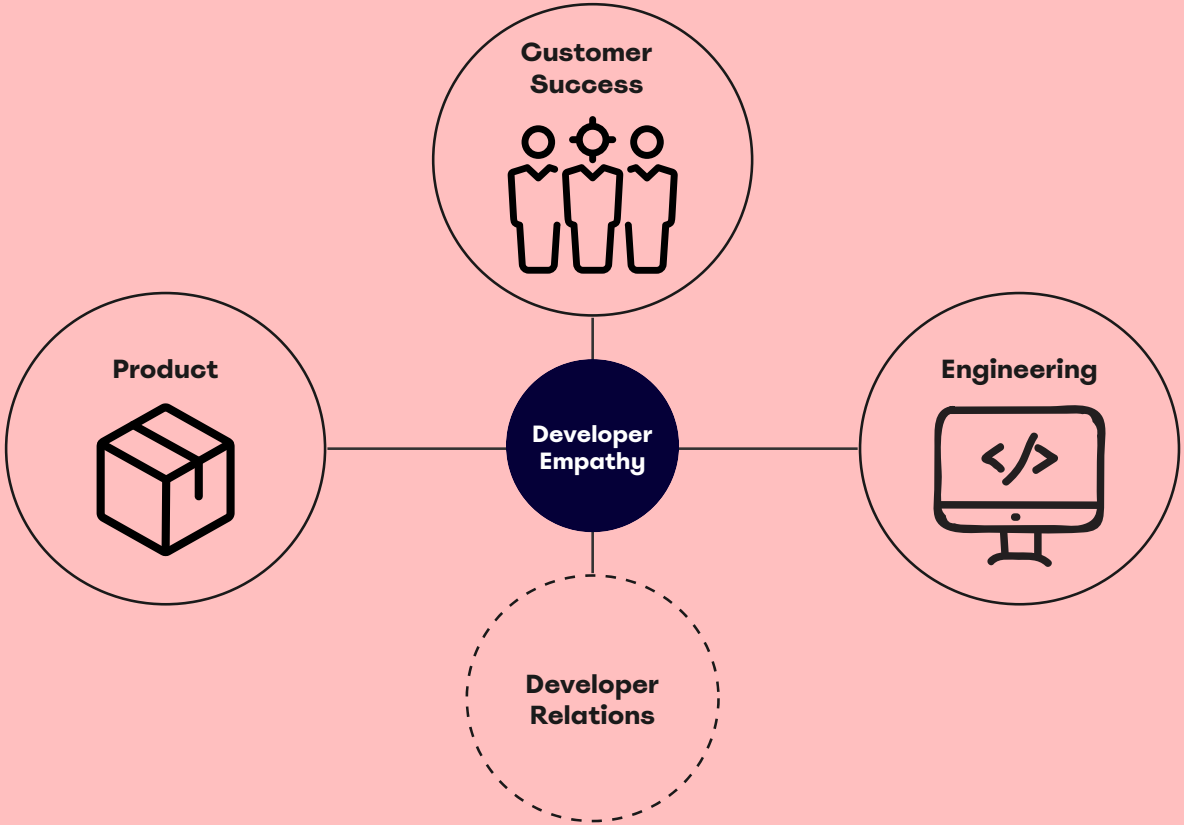
Deprecation Policy

And you get only one shot!

Your APIs  Developers

Botched Policy
(or no policy)


Execution


Customer Success
Product
Developer Empathy
Engineering
Developer Relations

miro

The Big 3: Timelines, dependencies, and messaging

1. **Achievable, proposed timeline** for MVP of your next API version
2. **Accounting for limitations** and dependencies
3. **A clear message** crafted for customers, with cross-functional buy-in

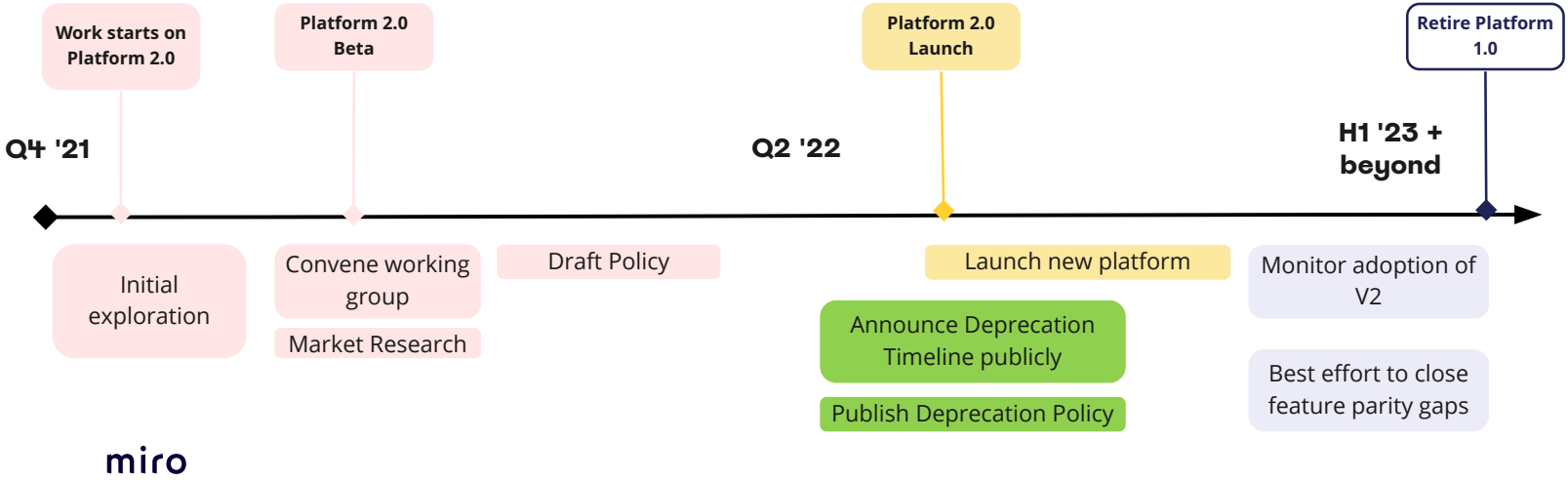
Understand the landscape

- What are **industry leaders** doing?
- How does your capability **fit into these standards**?
- Where do you **need to deviate**?

Case Study: Deprecating Miro Developer Platform 1.0

Miro Case Study

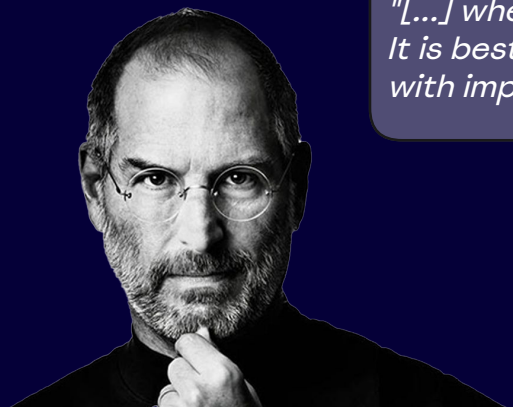
Case Study: Deprecating Miro Developer Platform 1.0



miro

Honesty and transparency above all

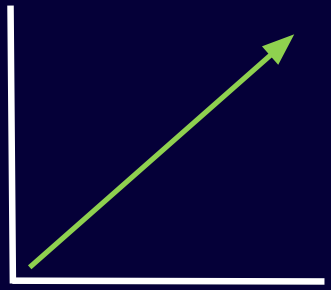
When developers choose to leverage your APIs, they're putting their faith in your product, and also your team's decision making.



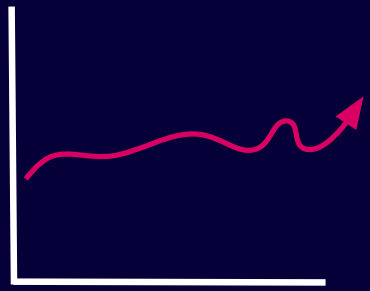
"[...] when you innovate, you make mistakes. It is best to admit them quickly, and get on with improving your other innovations."

Remain nimble and flexible where necessary

If: Desired Adoption



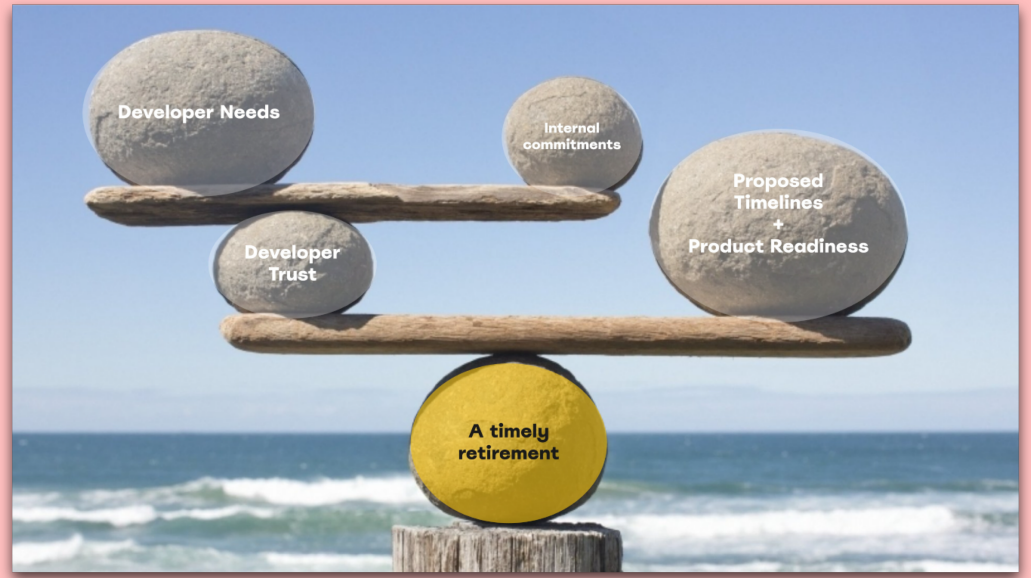
But: Actual Adoption



Then...

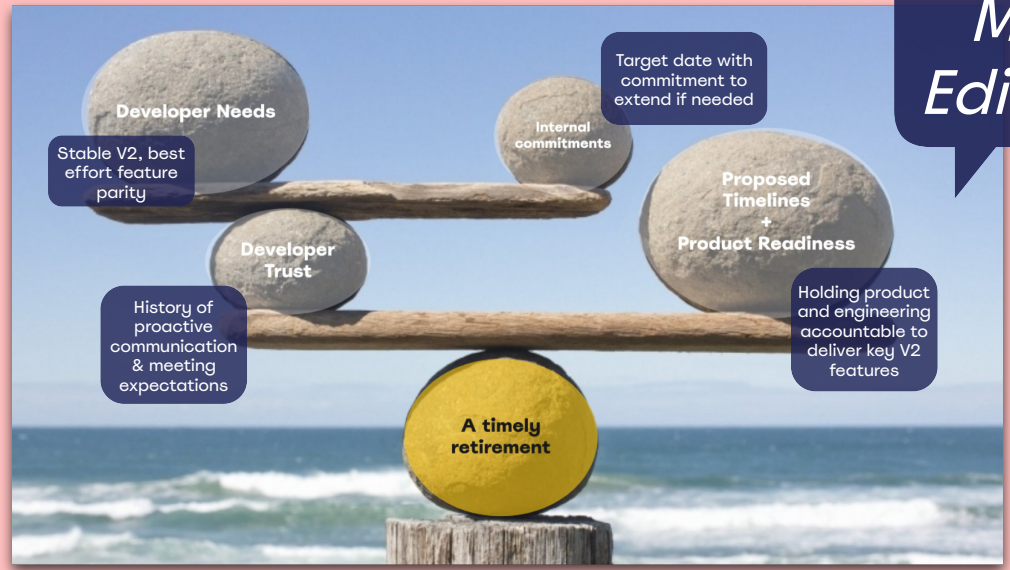

PIVOTS AHEAD

Balancing developer focus and commitment



miro

Balancing developer focus and commitment

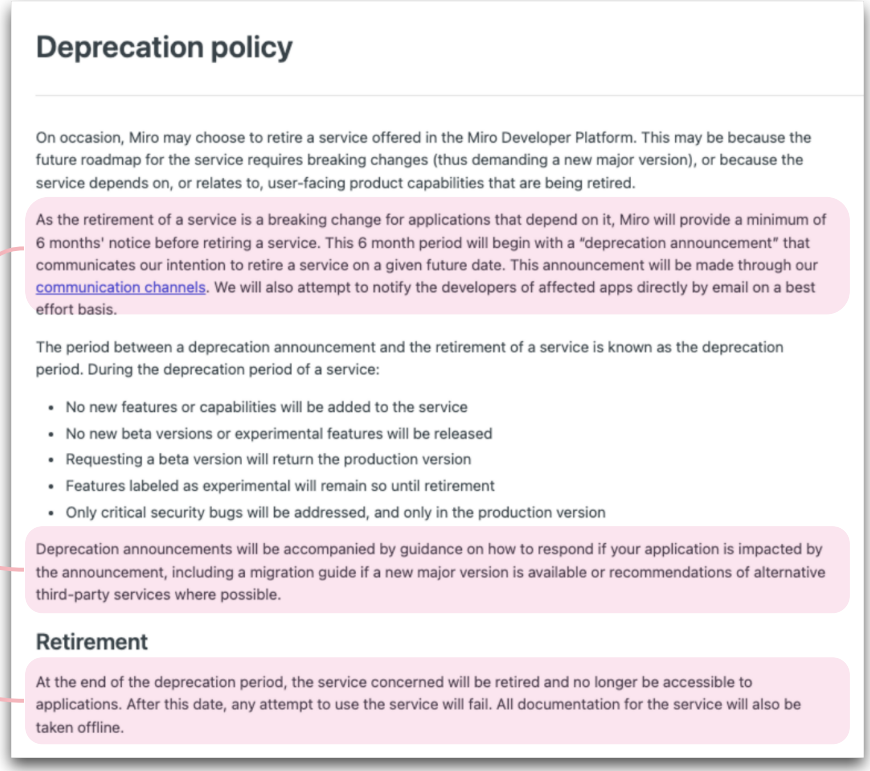


Miro Edition!

Miro Deprecation Policy

A policy that includes 3 critical components

1. **Concrete timeline** and communication strategy
2. **A commitment to developers** that they're not alone, with resources to help
3. **A clear definition** of deprecation and retirement phases



Deprecation policy

On occasion, Miro may choose to retire a service offered in the Miro Developer Platform. This may be because the future roadmap for the service requires breaking changes (thus demanding a new major version), or because the service depends on, or relates to, user-facing product capabilities that are being retired.

As the retirement of a service is a breaking change for applications that depend on it, Miro will provide a minimum of 6 months' notice before retiring a service. This 6 month period will begin with a "deprecation announcement" that communicates our intention to retire a service on a given future date. This announcement will be made through our communication channels. We will also attempt to notify the developers of affected apps directly by email on a best effort basis.

The period between a deprecation announcement and the retirement of a service is known as the deprecation period. During the deprecation period of a service:

- No new features or capabilities will be added to the service
- No new beta versions or experimental features will be released
- Requesting a beta version will return the production version
- Features labeled as experimental will remain so until retirement
- Only critical security bugs will be addressed, and only in the production version

Deprecation announcements will be accompanied by guidance on how to respond if your application is impacted by the announcement, including a migration guide if a new major version is available or recommendations of alternative third-party services where possible.

Retirement

At the end of the deprecation period, the service concerned will be retired and no longer be accessible to applications. After this date, any attempt to use the service will fail. All documentation for the service will also be taken offline.

If you remember nothing else, remember...

Good policy requires a **cross-functional** approach

Developers are **at the heart** of your decision-making

Invest time and resources **early on**

Questions

Further learning

Related topic:
API-as-a-Product eBook by Nordic APIs

How-To:
Best practices for deprecating APIs, by the trusted folks at Nordic APIs

How-To:
Compelling blog post on deprecating API endpoints by the Stoplight.io team

Thank you

Leave me feedback!